Detecting Guilty Party Using Dynamic Agents In Data Leakage

Lipi George, Mr.S.Vinoth Kumar, Dr.S.Karthik

ABSTRACT - *In today's business world, sometimes sensitive data must be handed over to supposedly trusted third parties. For example, a hospital may give patient records to researchers who will devise new treatments. Similarly, a company may have partnerships with other companies that require sharing customer data. Another enterprise may outsource its data processing, so data must be given to various other companies. We call the owner of the data the distributor and the supposedly trusted third parties the agents. Our goal is to detect when the distributor's sensitive data have been leaked by agents, and if possible to identify the agent that leaked the data. Here, we develop a model for assessing the "guilt" of agents. We also present algorithms for distributing objects to agents, in a way that improves our chances of identifying a leaker. We also consider the option of adding "fake" objects to the distributed set. Such objects do not correspond to real entities but appear realistic to the agents. In a sense, the fake objects act as a type of watermark for the entire set, without modifying any individual members. If it turns out that an agent was given one or more fake objects that were leaked, then the distributor can be more confident that agent was guilty.*

KEYWORDS - *Data Allocation Strategies, Data Leakage, Data Leakage Model, Data Privacy, Fake Objects, Guilty Agents, Sensitive Data.*

1. INTRODUCTION

A data distributor gives sensitive data to third parties. The data seems to be leaked and found in unauthorized places like on somebody's laptop or on the web. The distributor must assess that the agents only leaked the data and the data not leaked by other means. For example, for devising new treatments patient records are handed over by the hospitals to researchers. Here the owner of the data is the hospitals and the supposedly trusted agents are the researchers. The goal is to find when the agent leaked the distributor's sensitive data and also to identify the more accurate agent who leaked the data.

Previously, watermarking technique is used to detect the leakage. By watermarking technique, a unique code is injected into all distributed copies. The leaker can be identified if that copy is found in the hands of unauthorized parties. Watermarks are a useful method but it modifies the original data. Also if the data recipient is malicious [1] the watermarks are destroyed. By Perturbation technique, before the data being handed over to the agents, the data are modified and also the data are made "less sensitive". For example, ranges are replaced by exact values.[6].

In today's digital economy at record rates, data enters and leaves cyberspace. An enterprise on a daily basis sends and receives millions of email messages and downloads, saves, and transfers thousands of files through various channels. Enterprises also hold sensitive data that customer, business partners, shareholders expect them to protect. Unknowingly, companies constantly becomes victim to large data loss, and the data leakages involving sensitive personal and corporate data.. Data loss could harm a company's reputation. Therefore, organizations should understand the sensitive data that they are holding, how it's being controlled, and how to prevent it from leakage.

In this paper, the unobtrusive techniques are used for detecting leakage of a *set* of objects or records. The following scenario: The distributor discovers some of the same objects in an unauthorized place which he has been given to agents. For example, the data found on a web site, or obtained through a legal discovery process. At this point the distributor can assess that the leaked data came from one or more agents, and not by other means. We implement and analyze a guilt model which detects the agents without modifying the original data using allocation strategies.

The guilty agent is one of the agents who leaks a portion of distributed data. For this, distribute the data to agents based on sample data request and explicit data request in order to increase the chances of detecting the guilty agents. The algorithms implemented using fake objects would attain the goal of detecting guilty agents. By minimizing the sum objective there is an increase in chance of detecting guilty agents.Also developed a framework for generating fake objects.

We start in Section 2 with problem definition. In Sections 3, we present strategies for data allocation to agents.

2. PROBLEM DEFINITION

A distributor has a set $T = \{t_{1,\dots\dots,}tm\}$ of valuable data objects. The distributor shares some of the objects with a set of agents $U_1, U_2, \dots, Un$, but does not want the objects be leaked to third parties. The objects in T could be of any type and size, e.g., they can be tuples in a relation, or relations in a database. An agent receives a subset of objects of T, either by a sample request or an explicit request:

Sample Request R_i = SAMPLE (T, m_i) = Any subset of m_i records from T can be given to U_i.
Explicit Request R_i= EXPLICIT ($T, cond_i$) = Agent U_i receives all T objects that satisfy cond_i.

2.1. Agent Guilt Model

Suppose that the distributor discovers that a set S of T has leaked after giving objects to agents. This shows that some third party called the target has been caught in possession of S. For example, this target may be displaying S on its web site, or perhaps as part of a legal discovery process, the target turned over S to the distributor. It is reasonable to suspect $U_{1,\dots\dots}U_n$ since the agents have some of the data. However, the agents can argue that they did not leak the data, and that the S data was obtained by the target through other ways.

If an agent gives one or more objects to the target then we can say that the agent is guillty. The event that agent is guilty for a given leaked set S is denoted by $G_i \mid S$. The next step is to find $Pr \{ G_i \mid S \}$, i.e., the probability that agent is guilty given evidence S. To compute the $Pr \{ G_i \mid S\}$, estimate the probability that values in S can be "guessed" by the target. For example, some of the objects in t are emails of individuals.

Conduct an experiment and tell a person to find the email of about 100 individuals, the person may sometimes only discover say 20, which leads to an estimate of 0.2. Call this estimate as p_t , the probability that object t can be guessed by the target. We use the probability of guessing to identify agents that have leaked information.

The two assumptions regarding the relationship among the various leakage events.

Assumption 1: For all $t, t^{1} \in S$ such that $t \neq t^1$ the provenance of t is independent of the provenance of t^{1}.

The term provenance in this assumption statement is the source of a value t *that* occurs in the leaked set. The source can be any of the agents who have t in their sets or can be the target itself.

Assumption 2: An object $t \in S$ can only be obtained by the target in one of two ways.

- A single agent U_i leaked t from its own R_i set,
- The target guessed t without the help of any of the n agents.

With the assumptions, to find the probability that an agent U_i is guilty given a set S, consider the target guessed with probability p and that agent leaks to S with the probability 1-p. First compute the probability that he leaks a single object t to S. To compute this, define the set of agents $V_t = \{U_i \mid t \pounds R_i \}$ that have t in their data sets.

Then using Assumption 2 and known probability p, we have

$$Pr\{some\ agent\ leaked\ t\ to\ S\} = 1-p \qquad \dots\dots (1.1)$$

Assuming that all agents that belong to V_t can leak t to S with equal probability and using Assumption 2 we obtain,

$$p_r\{U_i\ \ leaked\ t\ to\ S\} = \begin{cases} \frac{1-p}{|V_t|}, & if\ U_i \in V_t \\ 0, & Oterwise \end{cases} \quad \dots\dots (1.2)$$

Given that agent is guilty U_i if he leaks at least one value to S, with Assumption 1 and Equation 1.2 compute the probability $Pr\{G_i \mid S\}$, agent U_i is guilty,

$$Pr\{G_i/S\} = \prod_{t \in S \cap R_i} \left(1 - \frac{1-p}{|V_t|}\right) \qquad \ldots\ldots(1.3)$$

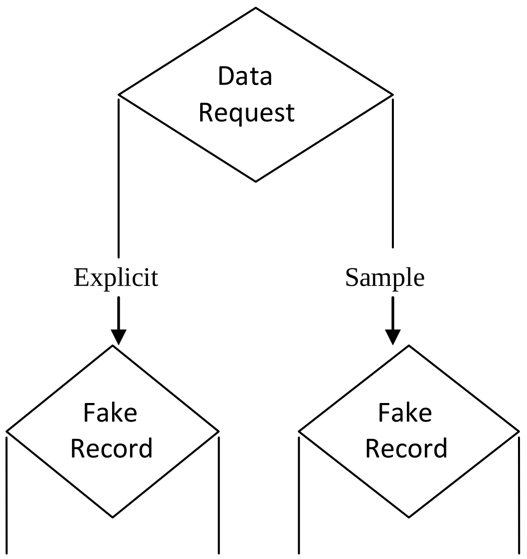
2.2. Data Allocation Problem

The main focus of our paper is the data allocation problem: how can the distributor "intelligently" give data to agents for improving the chances of detecting a guilty agent. There are four instances of this problem, depending on the type of data requests by the agents and whether "fake objects" are allowed[3].

Agent makes two types of requests, called sample request and explicit request based on the requests the fakes objects are added to the data list. Fake objects are objects generated by the distributor that are not in set *T*. The objects are designed to look like real objects, and are distributed to agents together with the *T* objects, for increasing the chances of detecting agents that leak data.

The Fig. 1 represents four problem instances with the names *EF, EF' , SF* and *SF'* , in which *E* stands for explicit requests, *S* for sample requests, *F* for the use of fake objects, and *F*' for the case where fake objects are not allowed.

i. Explicit request with fake tuples
ii. Explicit request without fake tuples
iii. Implicit request with fake tuples
iv. Implicit request without fake tuples.



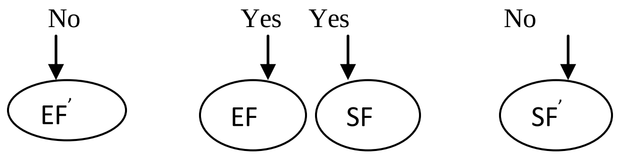


Fig 1: Leakage Problem Instances

Fake objects are the objects that appear as real to agents but do not correspond to real entities. The fake objects act as a type of watermark for the entire set, without making changes in any individual members. The distributor add fake objects to each of the distributed data for improving his effectiveness in detecting guilty agents.

The distributor creates and adds fake objects to the data which he distributes to agents. Fake objects must be created in such a way that the agents cannot distinguish them from real objects. The distributor must limit the number of fake objects received by each agent for the purpose of not to arouse suspicions. Thus, we say that the distributor can send up to *bi* fake objects to agent *Ui*.

The creation of a fake object for agent *Ui* is a black box function CREATEFAKEOBJECT (*Ri, Fi, condi*) that takes as input the set of all objects *Ri*, the subset of fake objects Fi that Ui has received, and condi, and returns a new fake object. The distributor can also use function CREATEFAKEOBJECT() when he wants to send the same fake object to a set of agents. In this case, the function arguments are the union of the *Ri* and *Fi* tables, respectively, and the intersection of the conditions *condi*.

In *EF* problems, the objective values are initialized by agents data requests. For example, $T = \{t_1, t_2\}$ and there are two agents with explicit data requests such that $R_1 = \{t_t, t_2\}$ and $R_2 = \{t_1\}$. The distributor cannot remove or alter the R_1 or *R2* data to decrease the overlap $R_1 \backslash R_2$. However, say the distributor can create one fake object (B = 1) and both agents can receive one fake object ($b_1 = b_2 = 1$). If the distributor is able to create more fake objects, he could further improve the objective of detecting the guilty agent.

3.DATA ALLOCATION STRATEGIES

3.1. Explicit Data Requests

In case of explicit data request with fake objects not allowed, the distributor is not allowed to add fake objects to each of the distributed data. So the Data allocation is fully defined by the data request of the agents.

In case of explicit data request with fake objects allowed, the distributor cannot remove or alter the requests R from the agent. However distributor can add the fake objects. In algorithm for data allocation for explicit request, the input to this is a set of request $R_1, R_2, \ldots\ldots R_n$, from n agents and different conditions for requests. The e-optimal algorithm finds the agents who are eligible to receiving fake objects. Then create one fake object in iteration and allocate that fake objects with the data to the agent selected. The e-optimal algorithm minimizes every term of the objective summation by adding maximum number b_i of fake objects to every set R_i yielding optimal solution.

In the first place, the goal is to see whether fake objects in the distributed data sets yield significant improvement in the chances of detecting a guilty agent. In the second place, evaluating e-optimal algorithm relative to a random allocation.

Step 1: Calculate total fake records as sum of fake records allowed.

Step 2: While total fake objects > 0

Step 3: Select agent that will yield the greatest improvement in the sum objective

$$i = argmax\left(\frac{1}{|R_i|} - \frac{1}{|R_i| + 1}\right)\sum_j R_i \cap R_j$$

Step 4: Create fake record

Step 5: Add this fake record to the agent and also to fake record set.

Step 6: Decrement fake record from total fake record set.

Algorithm makes a greedy choice by selecting the agent that will yield the greatest improvement in the sum-objective. In [4], we provide an algorithm that implements this intuition and we denote it by S-Sum.

3.2. Sample Data Request

With sample data requests, agents are not interested in particular objects. Hence object sharing is not explicitly defined by their requests. The distributor is forced to allocate certain objects to multiple agents only if the number of requested objects exeeds the number of objects in set *T*. The more data objects the agents request in total, the more recipients on average an object has; and the more objects are shared among different agents, the more difficult it is to detect a guilty agent.

Here, each agent U_i may receive any *T* from a subset out of $\binom{|T|}{m}$ different ones. Hence, there are different allocations. In every allocation, the distributor can permute *T* objects and keep the same chances of guilty agent detection. The reason is that the guilt probability depends only on which agents have received the leaked objects and not on the identity of the leaked objects. Therefore, from the distributor's perspective there are different allocations. An object allocation that satisfies requests and ignores the distributor's objective is to give each agent a unique subset of *T* of size *m*. The s-max algorithm allocates to an agent the data record that yields the minimum increase of the maximum relative overlap among any pair of agents[4]. The s-max algorithm is as follows.

```
Step 1: Initialize $Min\_overlap \leftarrow 1$, the minimum out of the maximum relative overlaps that the allocations of different objects to $U_i$

Step 2: for $k \in \{k \mid t_k \in R_i\}$ do
        Initialize $max\_rel\_ov \leftarrow 0$, the maximum relative overlap between $R_i$ and any set $R_j$ that the allocation of $t_k$ to $U_i$

  Step 3: for all $j = 1,\ldots, n : j = i$ and $t_k \in R_j$ do
        Calculate absolute overlap as
        $abs\_ov \leftarrow |R_i \cap R_j| + 1$
        Calculate relative overlap as
        $rel\_ov \leftarrow abs\_ov / min\ (m_i, m_j)$

 Step 4: Find maximum relative as
        $max\_rel\_ov \leftarrow MAX\ (max\_rel\_ov, rel\_ov)$
        If $max\_rel\_ov \leq min\_overlap$ then
        $min\_overlap \leftarrow max\_rel\_ov$
        ret_$k \leftarrow k$
        Return $ret\_k$
```

It can be shown that algorithm s-max is optimal for the sum-objective and the max-objective in problems where $M \leq |T|$ and $n < |T|$. It is also optimal for the max-objective if $|T| \leq M \leq 2\,|T|$ or all agents request data of the same size. It is observed that the relative performance of algorithm and main conclusion do not change. If p approaches to 0, it becomes easier to find guilty agents and algorithm performance converges. On the other hand, if p approaches 1, the relative differences among algorithms grow since more evidence is need to find an agent guilty.

The algorithm presented implements a variety of data distribution strategies which improves the distributor's chances of detecting a guilty agent. Distributing objects judiciously can make a significant difference in identifying guilty agents, especially in cases where there is large overlap in the data that agents must receive.

4. CONCLUSIONS

In a perfect world, handing over sensitive data to the agents may lead to unknown or malicious leakage of data. Watermark each of the data object so that it's possible to trace its origins clearly. However, in most cases we may work with agents that are not 100% trusted, and we may not be clear if a leaked object came from an agent or from some other means. In spite of these difficulties, it is possible to assess that the an agent is the leaker, based on the overlap of his data with the leaked data and the data of other agents, and also based on the probability that the objects can be guessed by other means.

The model is simple. The algorithms implement a variety of data distribution strategies that improves the distributor's chances of identifying a leaker. Distributing objects can make a significant difference in identifying guilty agents, in cases where there is large overlap in the data that agents must receive. The future work includes the extension of the allocation strategies to allocate data for online agents. Online agents can register and get the data from distributors. The Leakage detection use HTML parser extracts the contents from the leaked web url and analysis which agent has leaked the data.

ACKNOWLEDGMENT

Students work is incomplete until they thank the almighty & his teachers. I sincerely believe in this and would like to thank Dr.S.Karthik, Head of the Department, Computer Science & Engineering, SNSCT, Coimbatore for her encouragement and the motivation to write this paper. Also Iam grateful to Mr.Vinoth Kumar, Prof.,(CSE), SNSCT, Coimbatore for guiding me in writing this paper.

REFERENCES

[1] R. Agrawal and J. Kiernan, "Watermarking relational databases", In VLDB'02: Proceedings of the 28th international conference on Very Large Data Bases, pages 155–166, VLDB Endowment, 2002.

[2] P. Bonatti, S. D. C. di Vimercati, and P. Samarati, "An algebra for composing access control policies", ACM Trans. Inf. Syst. Secur., 5(1):1–35, 2002.

[3] S.U. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani, "Towards Robustness in Query Auditing, " Proc. 32nd Int'l Conference Very Large Data Bases (VLDB '06), VLDB Endowment,

pp. 151-162, 2006.

[4] P. Papadimitriou and H. Garcia-Molina, "Data Leakage Detection,"technical report, Stanford Univ., 2008.

[5] P. Papadimitriou and H. Garcia-Molina, "Data leakage detection," IEEE Transactions on Knowledge and Data Engineering, pages 51-63, volume 23, 2011.

[6] L.Sweeney,"Achieving K-Anonymity Privacy Protection Using GeneralizationandSuppression,"http://en.scientificcommons.org/43196131, 2002.